Implementación de un mecanismo de despliegue automático de servicios convergentes en entornos JSLEE

Implementation of an automatic deployment mechanism of convergent services on jslee environment

Julian A. Caicedo, M.Sc (c)
jacaicedo@unicauca.edu.co

Juan Carlos Corrales, Ph.D
jcorral @unicauca.edu.co

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Grupo de Ingeniería Telemática
Universidad del Cauca
Popayan-Colombia

Fecha de recepción: Febrero 18 de 2014
Fecha de aceptación: Marzo 28 de 2014

Palabras clave
Servicios convergentes; JSLEE; Mecanismo despliegue automático; redes de Petri coloreadas.

Keywords
Convergent Services, JSLEE, Automatic Deployment Mechanism, Coloured Petri Nets

Colciencias tipo 1

Resumen

En la actualidad los operadores de telecomunicaciones han incrementado el uso de recursos provenientes de diferentes fuentes, tales como facilidades de la web, contenidos generados por diversos proveedores, aplicaciones y servicios de terceros. Esta convergencia de servicios, aplicaciones y dispositivos ha propiciado en el operador telco la necesidad de contar con entornos para la entrega de servicios y de estrategias que faciliten el rápido despliegue de servicios convergentes [CS]. En este sentido, el presente artículo propone un mecanismo de despliegue de CS en entornos JSLEE, el cual, de manera automática, ejecuta los procesos de *activación*, *configuración*, *selección e instalación* del servicio. Con el fin de evaluar experimentalmente esta aproximación –y presentar sus aportes–, se realizó una prueba de desempeño del algoritmo propuesto.

Abstract

Nowadays, telecom operators are drawing on resources from web, content provider and third party applications. This convergence on services, applications and devices has carried to Telco operator to consider convergent service [CS] provision platforms that making deployment process faster and easier. Hence, this paper proposes a CS deployment mechanism in JSLEE environments which executes in an automatic way the activation, configuration, selection and installation process on the service. In order to experimentally assess our mechanism, we made a performance test with intent to illustrate ours contributions.

Agradecimientos

Al Departamento Administrativo de Ciencia, Tecnología en Innovación (Colciencias) y a la Universidad del Cauca por su soporte a este proyecto bajo el programa de Jóvenes Investigadores e Innovadores, y a los miembros del Grupo de Ingeniería Telemática [GIT] que participaron en la realización del proyecto TelComp 2.0.

I. Introducción

En la última década, los servicios de telecomunicaciones han incrementado el uso de recursos provenientes de diferentes partes, tales como la web, los proveedores de contenido, los proveedores de aplicaciones, y los servicios terceros y del mismo operador (Adell, 2006). En este sentido, existe un conjunto de servicios denominados servicios convergentes [CS] (Chudnovskyy, Weinhold, Gebhardt, & Gaedke, 2011), los cuales tienen la capacidad de combinar recursos, tanto del dominio Web, como del dominio Telco; De hecho, para que los usuarios puedan consumir un servicio convergente, es necesario pasar por los procesos de diseño/desarrollo, despliegue y operación (fases de aprovisionamiento). En cada una de estas fases se han planteado, desde una perspectiva académica y empresarial, diversas aproximaciones y estudios de importancia. Precisamente, en las dos primeras fases los autores definen la composición de servicios, enfocándose en la construcción de herramientas y ambientes de creación (Gonçalves da Silva, Ferreira Pires, & van Sinderen, 2011) o en técnicas de recuperación y composición dinámica, haciendo énfasis en alternativas semánticas (Bo et al., 2010). Por su parte, la fase de operación está orientada al desarrollo de Sistemas de Operación y Soporte [OSS].

El presente artículo se enfoca en la fase de despliegue de servicios convergentes, haciendo uso de un ambiente de ejecución como *JAIN Service Logic Execution Environment* [JSLEE], el cual se caracteriza por su fácil crecimiento, su alta disponibilidad, la gestión para el control del estado de los servicios, y por ser un entono de ejecución suficientemente robusto y con altos niveles de desempeño (OpenCloud, 2009), todos ellos, factores críticos para un operador de telecomunicaciones. Adicionalmente, el modelo de compontes y adaptadores de recursos de JSLEE permite la implementación de servicios de telecomunicaciones básicos (e.g., SMS, MMS, llamada, video-llamada) combinados con servicios de la Web (e.g. mapas, estado del clima, planificación de viajes, reservación de tiquetes, etc.), los cuales pueden ser enriquecidos con características de Web social; un ejemplo es el servicio *LinkedInJobNotificator*, el cual envía una oferta de trabajo, actualmente registrada en el servicio de *LinkedIn* del usuario, a su red social twitter, al correo personal y al celular, a través de una llamada de voz, transformando el texto de la oferta de trabajo en un archivo de audio.

El proceso de despliegue de servicios convergentes presentado en este trabajo, expone un conjunto de procesos genéricos que son implementados en un algoritmo automático con base en la teoría de Redes de Petri Coloreadas [CPN] y adaptado a la estructura de ejecución de JSLEE.

El resto del trabajo ha sido divido en las siguientes secciones. La sección II describe los estudios más relevantes en el despliegue de servicios; la sección III, presenta la metodología seguida en el trabajo y las consideraciones de diseño en la construcción

del mecanismo de despliegue automático de CS; la sección IV expone los resultados obtenidos de la evaluación del desempeño del mecanismo, y la sección V esboza algunas conclusiones y trabajos futuros.

II. Trabajos relacionados

La convergencia de servicios de telecomunicaciones ha sido abordada desde los dominios Web y Telco. A continuación se presentan los principales trabajos y estudios para la fase de despliegue de servicios compuestos (Web y Telco) como punto inicial para la definición de una estructura de despliegue de servicios convergentes.

A) Despliegue de servicios compuestos en el dominio Web

En este dominio la tendencia más significativa y de mayor interés para la comunidad empresarial (Microsoft Popfly, Yahoo! Pipes, Google GMasEd), es el uso de los Mashups. Esta tecnología no es más que una aplicación Web que realiza la interconexión de servicios remotos disponibles en la red (Sánchez et al., 2009). La creación de nuevos servicios basados en Mashups, necesita que el flujo de control de la estructura del servicio compuesto se despliegue sobre un ambiente estrictamente diseñado para la ejecución de scripts (Sánchez et al., 2009).

Riabov, Bouillet, Feblowitz, Liu, y Ranganathan (2008); Bozzon, Brambilla, y Michele Facca (2009); Shevertalov y Mancoridis (2008), exponen el uso de Mashups como creación y posterior despliegue en ambientes de ejecución para JavaScript. Un claro ejemplo de un ambiente de ejecución y despliegue basado en esta tecnología es WSo2 server (WSO2, 2011).

Proyectos, herramientas y plataformas como Intel Mash Maker, JackBe Presto, IBM Mashup Center, MyCocktail (Iglesias, Fernandez-Villamor, del Pozo, Garulli, & Garcia, 2010), ServFace (Feldmann et al., 2009), Karma (Tuchinda, Szekely, & Knoblock, 2008), CRUISe (Pietschmann, Voigt, Rümpel, & Meißner, 2009), MashArt (Daniel, Casati, Benatallah, & Shan, 2009), y Mashlight (Baresi & Guinea, 2010), se centran en la creación y el despliegue de servicios compuestos por medio de mashups.

Por otro lado, están tecnologías como la Edición Empresarial de Java [JEE] y el Bus de Servicios Empresariales [ESB], las cuales permiten el alojamiento y la coordinación de servicios basado en eventos. La primera, ya bien conocida por la comunidad empresarial, para el despliegue de servicios directamente relacionados al mundo web (Oracle, 2012); La segunda, orientada a la construcción de servicios empresariales, por medio de la coordinación de mensajes entre servicios de distintos dominios de aplicación (Tao & Wu, 2010). En ese sentido, Bo et al.(2010), Koning, aiSun, Sinnema, y Avgeriou (2009), proponen que la estructura de ejecución del servicio compuesto sea desplegada en un motor de Lenguaje de Ejecución de Procesos de Negocio [BPEL] (e.g., ActiveBPEL), o controlada por un framework soportado en la tecnología ESB.

Gonçalves da Silva, Ferreira Pires, y van Sinderen (2011), y Zhu, Zhang, Cheng, Wu, y Chen, (2011), presentan la ejecución de servicios híbridos (no completamente

convergentes), donde la coordinación de los servicios que conforman la estructura del servicio compuesto se hace a través de mensajes. Este tipo de propuestas no proporciona un control total sobre dicha estructura.

En la mayoría de las aproximaciones expuestas para este dominio, los servicios que componen la estructura final –la cual va a ser desplegada– solo consume recursos y capacidades Web, dado que el proceso de despliegue es ampliamente conocido y relativamente sencillo. No obstante, al tener en cuenta capacidades del mundo Telco, necesarias para formar una estructura compuesta convergente, el despliegue deja de ser un proceso meramente técnico, para convertirse en un proceso complejo y riguroso, donde el manejo de requisitos de alto rendimiento es indispensable para el despliegue y la posterior ejecución del servicio convergente.

B) Despliegue de servicios compuestos en el dominio Telco

La tendencia en los últimos cinco años se centra en la obtención de una plataforma de aprovisionamiento de usuario final, que opere del mismo modo que el modelo web 2.0 (Sánchez et al, 2009). En consecuencia, los operadores de telecomunicaciones empiezan a adoptar –y adaptar– las buenas prácticas de creación de servicios del dominio web, al dominio de las telecomunicaciones. Un claro ejemplo de esto es el continuo esfuerzo de las empresas del sector, por exponer sus recursos y sus capacidades de red a terceros. Por su parte, desde 1998 la industria viene desarrollando tecnologías que faciliten la creación de nuevos servicios de telecomunicaciones, como OSA/Parlay, Parlay X, JAIN, OneAPI, entre otras.

No obstante, desplegar estos servicios requiere de una plataforma eficiente y orientada a este dominio específico. En consecuencia, JSLEE, muestra mayores ventajas para ser considerada como tecnología principal en la ejecución de servicios de telecomunicaciones (Chrighton, Long, & Page, 2007). En ese sentido, Yelmo, del Álamo, Trapero, y Martín, (2011), Eichelmann, Fuhrmann, y Ghita (2010), Falcarin (2009), y Femminella, Maccherani, y Reali (2010), proponen enfoques con base en esta tecnología para el despliegue de servicios convergentes. De igual manera, los trabajos más significativos para este dominio provienen de proyectos como TeamCom (2001), Spice (2008), y Omelette (2011). Este último, es un caso típico de un proyecto Telco que adopta las prácticas web, especialmente en la aplicación de la tecnología Mashups y RESTful para la creación y el despliegue de servicios convergentes (Omelette, 2011). Por otro lado, se plantea que la creación, despliegue y ejecución de servicios convergentes se realice teniendo como centro de ejecución a un motor BPEL (Shin, Yu, Chung, & Kim, 2008; Yu et al., 2009), solución ineficiente para soportar los altos requisitos de un servicio Telco como son: alta disponibilidad, alto desempeño, manejo multiprotocolo, fiabilidad y comunicación en tiempo real.

Como conclusión de esta sección se puede reesaltar que el modelo petición-respuesta implícito en el modelo de servicios web compuestos, protocolos heterogéneos, comunicación asíncrona y requerimientos de tiempo real provenientes del mundo Telco,

son los elementos que deben ser considerados por el proceso de despliegue automático de servicios convergentes.

III. Metodología

La Figura 1 define las etapas empleadas en la construcción del proceso de despliegue automático de servicios convergentes en un entorno JSLEE. La primera, *Definición Elementos Despliegue*, define los procesos genéricos que están presentes en la fase de despliegue de servicios convergentes; la segunda, *Estructura Despliegue*, contiene el diseño de la estructura de despliegue de servicios convergentes; la tercera, *Implementación Mecanismo Despliegue Automático*, define el mecanismo que, de manera automática, emplea la estructura de despliegue para realizar el proceso de activación de un servicio convergente; la cuarta, *Validación*, realiza la verificación del mecanismo, con base en la prueba de desempeño. La etapa *Validación* realimenta a la etapa *Estructura Despliegue* con el objetivo de ajustar la estructura del CS. Cada una de las etapas definidas responde a los principios establecidos por el modelo integral para un profesional en ingeniería, específicamente, el Modelo para la investigación Documental [MID] y el Modelo para la Investigación Científica [MIC] (Serrano, 2008).

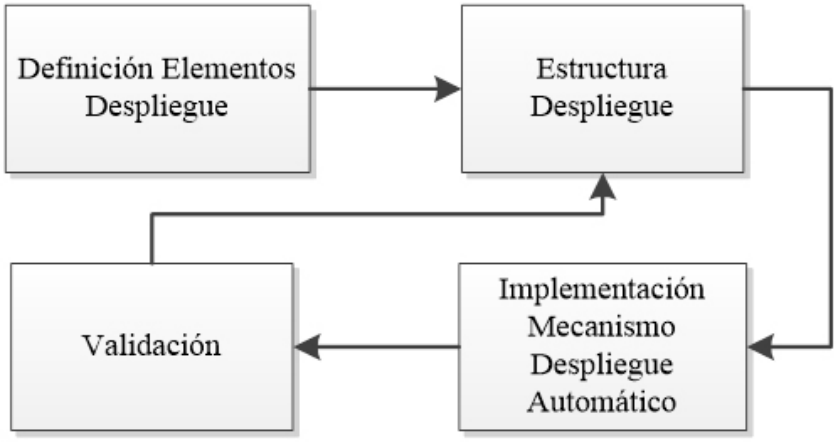

Figura 1. Diagrama de bloques en la construcción del proceso de despliegue automático de servicios convergentes en entornos JSLEE

La tecnología JSLEE es una especificación diseñada y optimizada para soportar aplicaciones conducidas por eventos, también conocidas como aplicaciones asíncronas (Sun Microsystems, 2008); sus principales elementos son el *modelo de componentes*, el *adaptador de recursos*, el *enrutador de eventos* y las *facilidades de gestión*. El primero es el elemento esencial de la arquitectura del SLEE y está estructurado por Bloques de Construcción [SBB], donde se describe la lógica funcional del servicio; el segundo, realiza la comunicación con entidades externas al SLEE a través de la adaptación de diferentes protocolos (*REST*, *SOAP*, *HTTP*, *Diameter*, *entre otros*); el tercero, enruta lo

eventos producidos al interior o exterior del SLEE hacia el componente lógico o SBB específico. Finalmente, las facilidades de gestión facilitan el control y el monitoreo de los SBB implementados al interior del SLEE (Sun Microsystems, 2008).

Teniendo en cuenta los elementos de JSLEE, específicamente su modelo de componentes y la estructura de empaquetamiento de la unidad desplegable [DU], la fase *Implementación Mecanismo Despliegue Automático* ha sido adaptada a la especificación.

A) Definición Elementos Despliegue

Esta etapa se enfocó en definir las *operaciones de invocación*, los *patrones de ejecución* y los *procesos genéricos* de despliegue para servicios convergentes, elementos necesarios y claves en el diseño de una estructura de despliegue. Para el primer componente, *operaciones de invocación*, se analizaron las características y las operaciones implícitas en los servicios Telco, a través de un descriptor estándar; en el segundo, p*atrones de ejecución*, se establecieron los principales patrones de flujo de control y flujo de datos empleados en los servicios; finalmente, el componente tres, *procesos genéricos*, describe aquellas acciones recurrentes en el proceso de despliegue de servicios de telecomunicaciones, las cuales fueron adaptadas para ejecutar el proceso de despliegue en servicios convergentes centrado en la tecnología JSLEE.

1. Operaciones de invocación

Los servicios que hacen parte de la estructura de un servicio convergente (Web-Telco) deben tener un descriptor que exponga las características, propiedades, ubicación, operaciones u otra información relacionada con su invocación. En ese sentido, los servicios Telco que componen el servicio deben estar representados por algún tipo de interfaz que permita identificarlos dentro del entorno de ejecución. Este trabajo no abordó la definición de un descriptor de servicio web en razón a la existencia de tecnologías claramente definidas y ampliamente utilizadas en la industria, como referentes en la descripción del servicio (WSDL-REST).

a. OSA/Parlay

Esta tecnología ha definido una arquitectura que hace posible el inter-funcionamiento entre las aplicaciones IT y las características de red de un operador de telecomunicaciones, a fin de construir servicios convergentes, tanto en redes fijas, como en móviles (Falcarin & Licciardi, 2003). Para brindar el acceso a los recursos del operador por terceros, OSA/Parlay definió un conjunto de API que encapsula las funciones de control y señalización usadas al interior de la infraestructura Telco. Estas interfaces gestionan el acceso a capacidades como: el establecimiento de llamadas de voz, la gestión de sesiones de datos, y la gestión de movilidad, mensajería y presencia (Zuidweg, 2009). Estas interfaces fueron definidas en el Lenguaje de Modelado Unificado [UML] y en el Lenguaje de Definición de Interfaz [IDL] dejando la implementación a criterio del desarrollador (Kryvinska, Strauss, Auer, & Zinterhof, 2008).

b. Parlay-X

Esta tecnología define un conjunto de servicios Web que brindan acceso abstracto,

tanto de comunicaciones basadas en SS7, como en redes basadas en SIP. El modelo seguido para el despliegue de estas interfaces se basa en el mecanismo tradicional de publicación de servicios web. Los servicios de Parlay-X son habilitados a través de sus interfaces publicadas en un registro e implementadas sobre el Parlay Gateway. Estas interfaces web no son más que un nivel de abstracción más alto que las API de OSA/Parlay. El mecanismo de autenticación y autorización sigue el mismo modelo de la arquitectura OSA/Parlay (Unmehopa, Vemuri, & Bennett, 2006).

Entre los servicios definidos por esta tecnología se encuentran: notificación de llamada, mensajería corta, mensajería multimedia, manejo de llamada, conferencia multimedia, entre otros. Las descripciones de estos servicios se encuentran definidos por sus WSDL o archivos XML, en algunos casos (3GPP, 2009).

c. OneAPI

Corresponde a un conjunto de API que expone capacidades de red sobre la Internet, las cuales forman parte de las API de red de OMA RESTFul. En ese sentido, la interfaz de descripción toma como base REST –para la comunicación del servicio– y JSON –para el intercambio de datos–.

A diferencia de Parly X, oneAPI permite un modelo de abstracción mayor para servicios o capacidades de red ofrecidos por el operador de telecomunicaciones, lo cual, en ocasiones, es más adecuado para aquellos desarrolladores del mundo IT. Este esquema de representación ha definido un conjunto de capacidades o servicios Telco, como mensajería multimedia, llamada tripartita, notificación de llamada, ubicación del terminal, presencia, entre otros (GSMA, 2012).

La Figura 2 ilustra la ruta de evolución de las interfaces de descripción de servicios Telco.

d. TelcoML

El Grupo de Gestión de Objetos [OMG] establece está especificación para la definición de un perfil UML en el diseño de servicios avanzados e integrados de

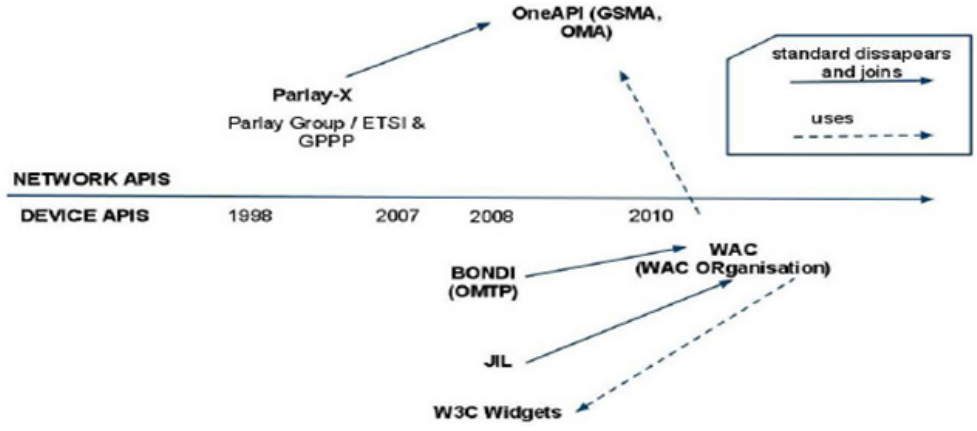

Figura 2. Evolución interfaces servicios Telco (Omelette, 2011)

telecomunicaciones, estandarizando la representación UML para un conjunto de interfaces de servicios Telco y la manera de representar su composición (OMG, 2012).

Esta especificación toma como base servicios Telco definidos por otros organismos –como OMA, GSMA y TM Forum–, entre ellos: mensajería corta, mensajería multimedia, *click to call*, gestión de ubicación de usuario, sincronización, reconocimiento de voz, y gestión de privacidad, entre otros.

Para cada uno de los servicios se ha defendido un conjunto de operaciones, parámetros de entrada, respuesta de invocación y condiciones de excepción de funcionamiento. En ese sentido, se estandariza la forma de comunicarse con los servicios de telecomunicaciones. Para este trabajo se seleccionó esta especificación debido a la definición estándar en los parámetros de los servicios Telco y al modelo orientado a servicios como SOA y SDP, conceptos fundamentales del mundo Web.

2. Patrones de ejecución

El flujo de ejecución está separado en: flujo de control y flujo de datos (van der Aalst, 2011). El primero, está relacionado con el orden de ejecución; el segundo, representa la transferencia de información entre servicios. En ese sentido, se tomó como referente a los trabajos de Benavides, Enríquez, Ramírez, Figueroa, y Corrales (2012), y van der Aalst (2011), para los patrones de control y los patrones de datos de servicios convergentes, respectivamente.

3. Procesos genéricos de despliegue

La Figura 3 ilustra los procesos genéricos para el despliegue de servicios convergentes (Kecskemeti, Terstyanszky, Kacsuk, & Nemétha, 2011).

El proceso de *selección*, escoge los objetos apropiados (software-hardware) para desplegar el servicio; el proceso de *activación* hace disponible el servicio para su ejecución; el proceso de *instalación*, gestiona la adición de los componentes software al ambiente de despliegue; el proceso de *configuración*, ajusta los componentes a las especificaciones del ambiente de despliegue; el proceso de *adaptación*, adapta los componentes software instalados y configurados previamente mientras el ambiente de despliegue está corriendo y recibiendo solicitudes; el proceso de *actualización*, remplaza

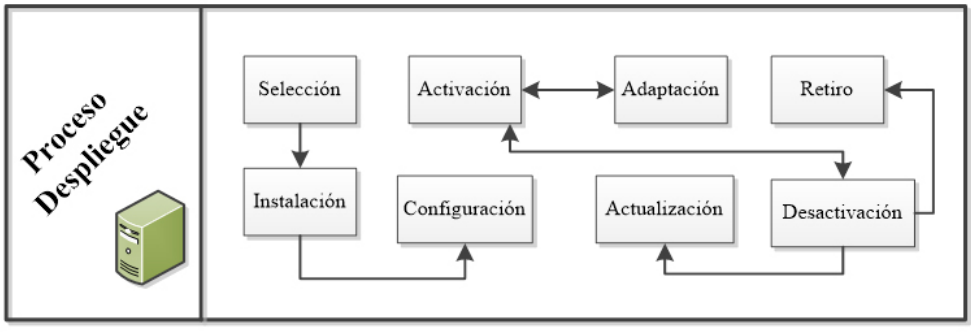

Figura 3. Procesos genéricos de despliegue

los componentes software previamente instalados y reinicia la configuración en ellos; el proceso de *desactivación*, hace uso de operaciones de mantenimiento para controlar aquellos módulos que no pueden operar en un servicio activo; y el proceso de *retiro*, remueve los componentes software del servicio del ambiente de despliegue.

Los componentes software están representados por los servicios básicos o atómicos (Web o Telco) implementados como SBB dentro del entorno JSLEE, mientras que el componente hardware es representado por el servidor que aloja dicha tecnología.

B) Estructura de despliegue

Esta etapa se enfocó en definir una estructura de despliegue para CS con base en los elementos establecidos dentro del proceso (*operaciones de invocación*, *patrones de ejecución y procesos genéricos*). Para ello, es esencial establecer un modelo formal que facilite el análisis, en conjunto, de estos tres elementos, obteniendo una estructura de despliegue coherente. En este sentido, las CPN fueron escogidas como el modelo más adecuado para la representación de SC, principalmente por el cumplimiento de criterios como: el *manejo del tiempo*, el *flujo de datos*, la *facilidad de composición*, las *herramientas y algoritmos*, y la *estandarización* (TelComp 2.0, 2013). Adicionalmente, este modelo está diseñado para describir sistemas concurrentes, asíncronos, distribuidos, paralelos, no deterministas y estocásticos (Jensen & Kristensen, 2009), características implícitas en el aprovisionamiento de servicios convergentes. De igual manera, las CPN añaden información a los servicios, definiendo un conjunto de *tokens* especiales, los cuales pueden contener información diferente, lo que implica que sean procesados de acuerdo con el dato representado por ellos.

Para cada servicio atómico, Web o Telco, la representación en CPN está definida por un lugar de entrada, *esperando invocación*, una transición intermedia, *ejecutar lógica del servicio*, y un lugar de salida, *enviando respuesta* (ver Figura 4). El punto en el lugar de entrada representa el tipo de dato (*Token*) que maneja el servicio atómico; el *token*

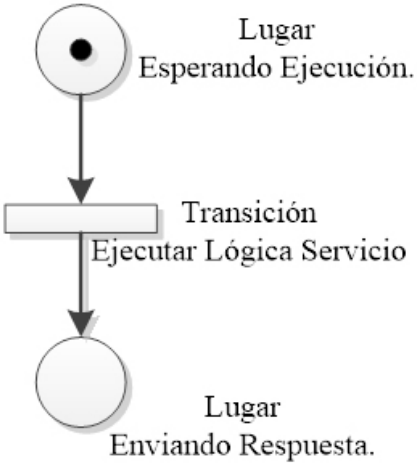

Figura 4. Representación de un servicio atómico Web o Telco en CPN

puede variar de color, dependiendo del dato que manaje el servicio; incluso, un servicio atómico pude manejar *n* datos, representados por *n* token de colores diferentes Estos servicios son representaciones de los servicios que cumplen con alguna meta de negocio dentro de la funcionalidad del servicio total.

El comportamiento interno del servicio podría estar descrito por una cantidad numerosa de estados y transiciones, lo cual haría complejo el análisis de su estructura. En ese sentido, se realiza la representación del servicio como una abstracción de su funcionalidad, manteniendo el requisito funcional del servicio.

Cada uno de los servicios atómicos se comunica a través de patrones de control como: secuencia, *parallel split* y *simple merge*, entre otros (Benavides, Enriquez, Ramirez, Figueroa, & Corrales, 2012). La Figura 5 ilustra la implementación en PN para los patrones mencionados, con base en lo expuesto por Russell, ter Hofstede, van der Aalst, y Mulyar (2006). Estos patrones de control representan la comunicación entre servicios y no el flujo de datos entre ellos (Figura 6); esto último se estructuró siguiendo la teoría de token de CPN.

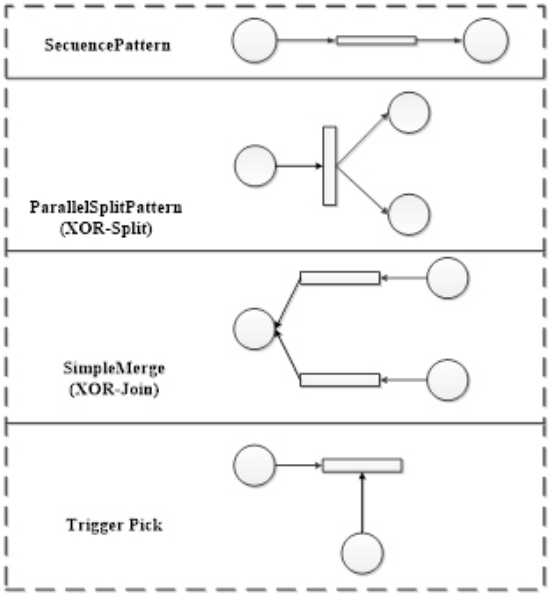

Figura 5. Representación de patrones en PN

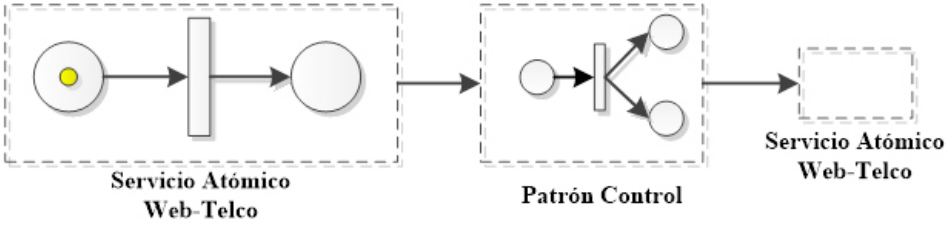

Figura 6. Conexión servicios atómicos

C) Implementación mecanismo despliegue automático

Esta etapa se enfocó en la construcción de un mecanismo de despliegue automático para CS con base en la estructura definida. Se implementó en tres fases: la primera, se encarga de generar los archivos ejecutables de la estructura del servicio; la segunda, realiza la configuración de despliegue en el entorno JSLEE; y la tercera, realiza la carga de la estructura del servicio convergente. Todas las fases hacen parte del siguiente proceso:

La información contendida en la estructura del servicio convergente es inferida a través de la herramienta *Petri Net Translator* (componente software desarrollado en el seno del proyecto TelCop2.0); posteriormente, un *generador de código* introduce los parámetros específicos de la tecnología JSLEE y genera, tanto los archivos fuente, como el archivo descriptor del servicio; el módulo *compilador* genera los archivos *class*; el componente de *configuración* introduce los parámetros de la estructura específicos para el despliegue y ejecución dentro de la tecnología JSLEE; finalmente, el componente de *carga del servicio* instala y activa el servicio en el servidor. La Figura 7 ilustra el proceso descrito.

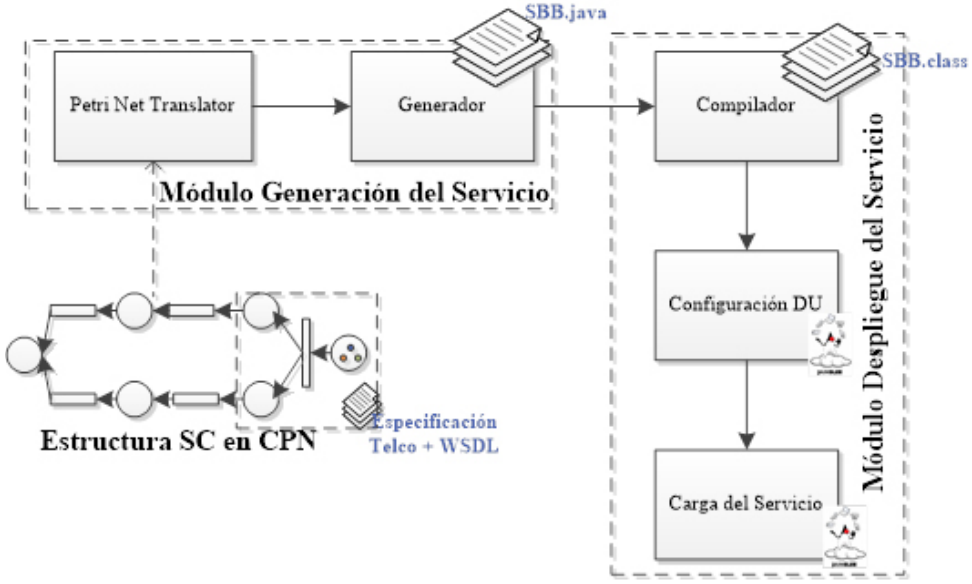

Figura 7. Mecanismo Generación-Despliegue automático para servicios convergentes en entornos JSLEE

El mecanismo ilustrado en la Figura 7 tiene dos módulos principales, el módulo de *generación del servicio* y el módulo de *despliegue del servicio*. En este trabajo, el módulo de *generación del servicio* está por fuera del alcance, sin embargo, ha sido utilizado como herramienta de entrada para el *módulo de despliegue*. A continuación se describen los módulos que componen el mecanismo de despliegue automático:

- *Compilador*. Este componente tiene como función revisar la sintaxis del código java y la estructura del servicio de acuerdo con la estructura definida por la tecnología JSLEE. El producto final es un *.class* que contiene la lógica funcional del servicio convergente.

- *Configuración DU*. Este módulo es el encargado de definir la estructura de despliegue del servicio convergente, en particular, la creación de la Unidad Desplegable [DU] y sus archivos de configuración.

La Figura 8 ilustra la estructura de una DU de un servicio en JSLEE.

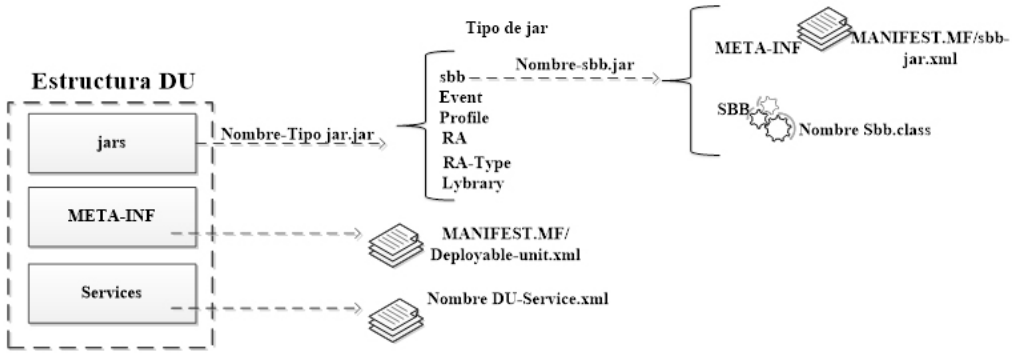

Figura 8. Estructura de despliegue de la Unidad Desplegable en entornos JSLEE

» El archivo *jars* contiene los elementos que interactúan en el funcionamiento del servicio; dependiendo de su complejidad, varios elementos pueden ser introducidos con el fin de resolver una actividad dentro de la funcionalidad del servicio. JAINSLEE ha definido un conjunto de elementos clave (i.e., *sbb*, *event*, *profile*, *RA type*, *RA y Lybrary*). El elemento principal es el sbb, en este *.jar* se encuentran alojados los *.class* del servicio, y los archivos de meta-datos y descriptor del servicio, MANIFEST.MF y .XML, respectivamente.

» El archivo *META-INF* contiene el descriptor de la DU (deployable-unit.xml) y el archivo meta-data (MANIFEST.MF)

» El archivo *services* contiene el descriptor del servicio convergente.

Carga del servicio. Módulo encargado de activar el servicio en el servidor para su posterior ejecución.

El módulo de despliegue automático implementa los procesos de *selección*, *activación*, *instalación* y *configuración*. La tarea de selección parte del código java del servicio y genera el archivo lógico ejecutable con las respectivas dependencias a otros servicios atómicos; cada uno de estos servicios son instancias software ejecutándose en el servidor; la tarea de activación e instalación se refleja en el despliegue de la DU en el servidor; finalmente, la tarea de configuración se implementa en la construcción de la DU del servicio que hace posible el despliegue efectivo en la tecnología JAINSLEE. La Figura 9 muestra las tareas que cada módulo del mecanismo de despliegue automático implementa.

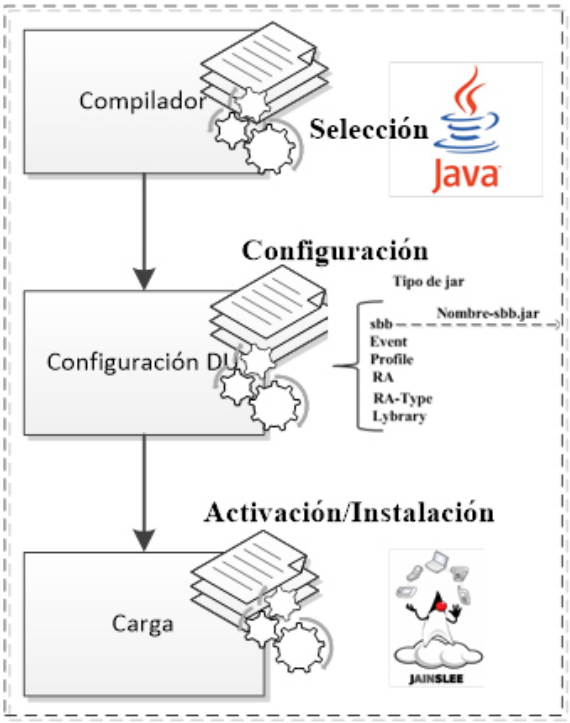

Figura 9. Implementación de los procesos de despliegue.

D) Validación del mecanismo despliegue

Esta etapa se enfocó en validar el mecanismo de despliegue automático de CS con base en los requerimientos no funcionales, centrándose en el consumo de los recursos computacionales (Kankanamge, 2012). Para ello, se realizó una prueba de desempeño midiendo las métricas de tiempo, CPU, memoria heap y CS exitosos –Femminella, et al. (2009), y Rojas-Sierra, Estrada-Solano, y Caicedo-Muñoz, (2011), definieron métricas similares en sus trabajos, para evaluar el desempeño y la escalabilidad de servicios en plataformas JSLEE–.

Teniendo en cuenta las ventajas de la Arquitectura Orientada al Servicio [SOA] y la implementación del mecanismo de despliegue automático sobre un servicio Web con uso de mensajes SOAP para el intercambio de los datos, se empleó la herramienta SOAPUI (SOAP UI, 2013) para la configuración de los parámetros de la prueba de estrés, la cual tiene como objetivo encontrar el punto de quiebre del sistema, determinando los límites de las métricas definidas. SOAPUI facilita la rápida creación de test avanzados de desempeño y la evaluación de servicios web (Hussain, Wang, Kalil, & Diop, 2013); además, es una herramienta de código abierto distribuida bajo la licencia GNU LGPL.

La prueba se realizó haciendo uso de dos computadoras personales [PC], de la herramienta SOAPUI y de una conexión de red entre los equipos. La Figura 10 ilustra el escenario de la prueba.

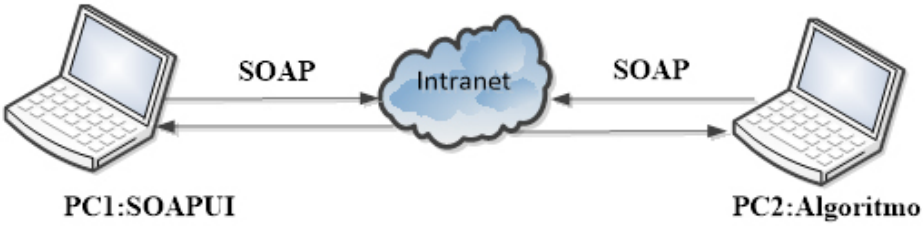

Figura 10. Escenario de prueba desempeño

IV. Resultados

Para la validación del mecanismo se diseñó el experimento de ruptura con base en la prueba de estrés. Este experimento determinó el punto de quiebre del algoritmo y los límites de la métricas definidas.

El punto de ruptura es el punto límite de carga donde el sistema colapsa (SOAP UI, 2013). En este sentido, se configuró la estructura del servicio convergente con *n* servicios atómicos, donde *n* varía de cuatro (4) en cuatro (4) hasta un límite de 12 servicios, y se aumentó el número de servicios convergentes que se enviaron de manera simultánea al algoritmo de despliegue automático. Los parámetros de configuración fueron los siguientes

- Número de hilos: *p*, donde $p = \{1, 2, 3,4,\ldots, k\}$
- Número de servicios atómicos (Web o Telco): *n*, donde $n = \{4, 8,12\}$

El *número de hilos* (*p*), hace referencia a la cantidad de servicios convergentes que se envían simultáneamente para ser procesados. La cantidad de éstos varió de uno (1) en uno (1) hasta un límite de k, donde k es el punto de ruptura del algoritmo.

El *número de servicios atómicos* (*n*) hace referencia a la cantidad de servicios que componen el servicio convergente. Para cada *n* servicios atómicos se determinó el punto de ruptura k. El límite de 12 servicios se seleccionó teniendo en cuenta la cantidad de servicios usados en el proceso de composición de servicios de Eichelmann, Fuhrmann, y Ghita (2010), y Gonçalves da Silva, Ferreira Pires, y van Sinderen (2011).

Las Figuras 11, 12,13 14, muestran los resultados obtenidos para este experimento con base en las métrica de: tiempo, *heap memory*, CPU y CS exitosos, respectivamente.

De la Figura 14 se obtiene que el punto de ruptura del algoritmo es de 5 CS, independiente de la complejidad de la estructura del servicio (4, 8 o 12 nodos). Esto se debe a la saturación que produce el algoritmo implementado en la herramienta *Petri net transator*, que genera un impacto final sobre el mecanismo automático de despliegue. Sin embargo, en la práctica, dos o más CS no serán enviados de manera simultánea para ser desplegados; el diseño y la construcción de un servicio, a través de un ambiente de creación [SCE], requiere de un análisis de los requerimientos de negocio, donde el tiempo consumido por éste difícilmente coincidirá con el tiempo de diseño, creación y despliegue de otro CS.

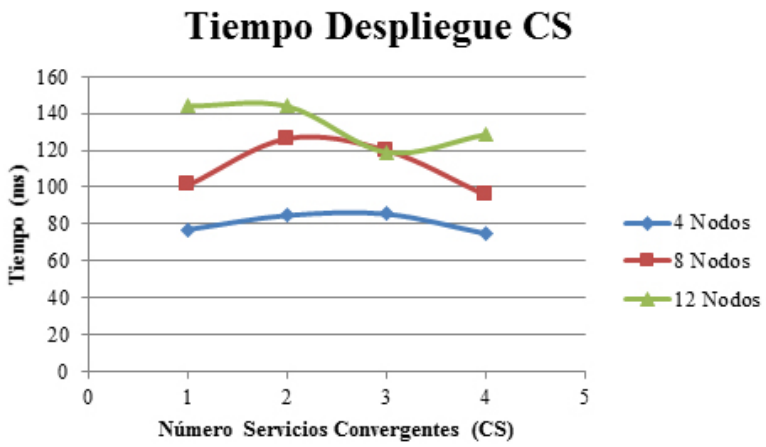

Figura 11. Tiempo despliegue automático

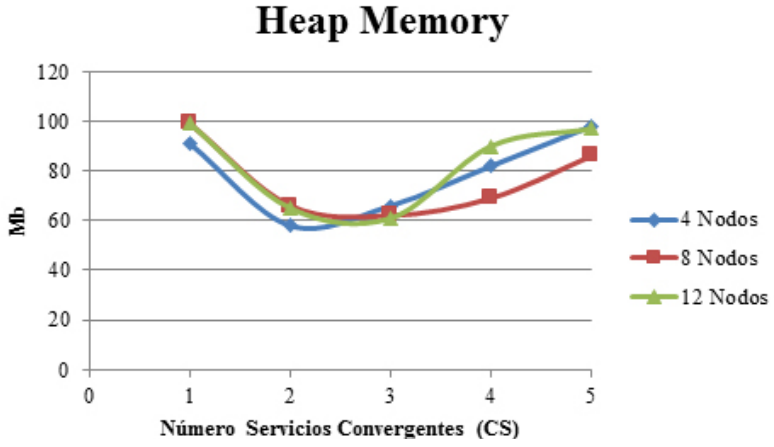

Figura 12. Memoria heap

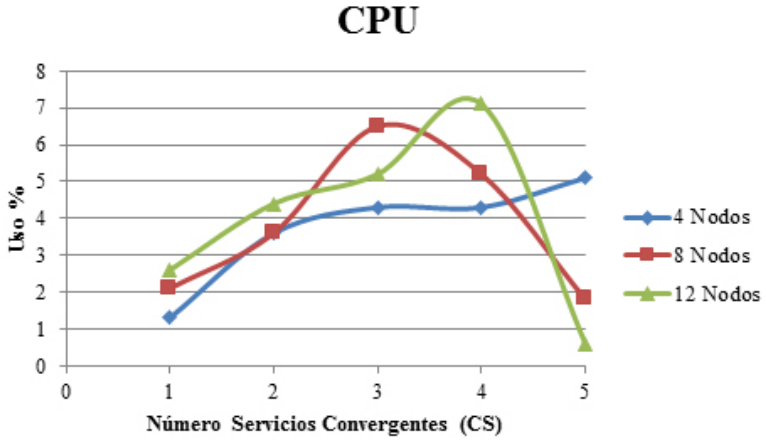

Figura 13. Uso CPU

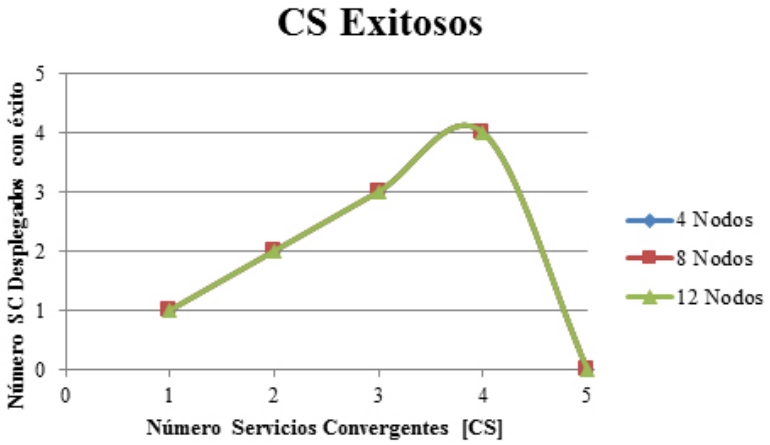

Figura 13. CS exitosos

De igual manera, en el punto de ruptura se presenta un consumo elevado en la memoria *heap* (Figura 12) y un bajo uso de CPU del sistema (Figura 13). Esto último se debe a que la Máquina Virtual Java [JVM] queda en estado inactivo, como producto del colapso del algoritmo de despliegue.

Adicionalmente, el consumo del recurso tiempo en el proceso de despliegue no supera los 160ms para 4 CS con 12 servicios atómicos enviados simultáneamente. En consecuencia, es posible inferir que el tiempo empleado por el proceso de despliegue automático será menor si se envían CS en secuencia.

Teniendo en cuenta a Eichelmann, Fuhrmann, y Ghita (2010), Gonçalves da Silva, Ferreira Pires, y van Sinderen (2011), los servicios que componen la estructura de un servicio compuesto no superan los 10 servicios dentro del dominio Telco. En este sentido, el algoritmo de despliegue automático tiene un comportamiento óptimo y eficiente para este intervalo; además, diseñar un servicio compuesto con gran cantidad de servicios atómicos, es en la práctica, una tarea compleja de realizar, principalmente, por la dificultad en el control de las interacciones de las entidades que forman parte del modelo de negocio del servicio.

Finalmente, es posible concluir que el comportamiento del algoritmo en el intervalo de 1 a 4 CS procesados simultáneamente con 12 servicios atómicos, es óptimo y eficiente; el consumo de tiempo, de memoria heap y de CPU muestra un uso adecuado de los recursos computacionales del sistema.

Conclusiones y trabajo futuro

Las CPN brindan un mayor poder de modelamiento en la estructura de despliegue de un CS, facilitando su manipulación, análisis e integración, específicamente por la expresividad que poseen para representar servicios compuestos en el dominio

de las telecomunicaciones. La separación de componentes lógicos (*compilador SBB*, *configuración DU y carga del servicio*) y de procesos genéricos de despliegue (*configuración*, *activación*, *instalación y selección*), y la implementación de algoritmos con base en paradigmas orientado a objetos y por eventos (*java y jslee*, *respectivamente*), permitieron definir un mecanismo de despliegue automático de CS dinámico, eficiente y efectivo, que podría ser empleado en procesos de composición de servicios asistidos por entornos gráficos.

El principal enfoque futuro de investigación se centra en estudios y desarrollos que incluyan los procesos de despliegue orientados a controlar el ciclo de vida de un servicio convergente, en particular, procesos de *adaptación*, *actualización*, *desactivación* y *retiro* serán claves para realizar la gestión del servicio en un entorno convergente JSLEE. S&T

Referencias bibliográficas

3GPP. (10 de diciembre de 2009). *3GPP Specification detail. Open Service Access (OSA); Parlay X web services; Part 1: Common*. Recuperado de http://www.3gpp.org/ftp/Specs/html-info/29199-01.htm

Adell, J. et al. (2006). *Telecomunicaciones de nueva generación*. Madrid, España: Telefónica.

van der Aalst, W. (2011). *Workflow Patterns Initiative*. Recuperado de http://www.workflowpatterns.com/

Baresi, L., & Guinea, S. (2010). Consumer mashups with mashlight. *Lecture Notes in Computer Science [Towards a service based Internet]*(6481), 112–123.

Benavides, A., Enriquez, G., Ramirez, J., Figueroa, C., & Corrales, J. (2012). Control-flow patterns in converged services. En A. Zimmermann (Ed.), *The Fourth International Conferences on Advanced Service Computing* (pp. 37-42). Red Hook, NY: Curran.

Bo, C., Yang, Z., Peng, Z., Hua, D., Xiaoxiao, H., Zheng, W., & Junliang, C. (Marzo de 2010). Development of Web-Telecom based hybrid services orchestration and execution middleware over convergence networks. Journal of Network and Computer Applications, *33*, 620-630.

Bozzon, A., Brambilla, M., & Michele Facca, F. (2009). A conceptual modeling approach to business service mashup development. 751-758.

Chrighton, C., Long, D., & Page, D. (Diciembre de 2007). JAIN SLEE vs SIP Servlet.Which is the best choice for an IMS aplication server? *Australasian Telecommunication Networks and Applications Conference*, (pp. 448-453).

Chudnovskyy, O., Weinhold, F., Gebhardt, H., & Gaedke, M. (2011). Integration of Telco Services into Enterprise Mashup Applications. En Springer (Ed.), Proceedings ICWE'11 Proceedings of the 11th international conference on Current Trends in Web Engineering.

Berlín, Alemania: Springer-Verlag.

Daniel, F., Casati, F., Benatallah, B., & Shan, M. (2009). Hosted Universal Composition: Models, Languages and Infrastructure in mashArt. *Lecture Notes in Computer Science*, (pp. 428–443).

Eichelmann, T., Fuhrmann, W., & Ghita, B. (2010). *Support of parallel BPEL activities for the TeamCom service creation platform for next generation networks.* Recuperado el 26 de Septiembre de 2010, de http://www.e-technik.org/aufsaetze_vortraege/aufsaetze/eichelmann_fuhrmann_trick_ghita_sein2009_vas.pdf

Falcarin, P. (2009). Service composition quality evaluation in SPICE platform. En *High assurance services computing* (págs. 89-102). New York, NY: Springer-Verlag.

Falcarin, P., & Licciardi, C. A. (2003). Analysis of NGN service creation technologies. *International Engineering Consortium (IEC) Annual Review of Communication*, *56*, pp. 537-551.

Feldmann, M., Nestler, T., Muthmann, K., Jugel, U., Hübsch, G., & Schill, A. (2009). Overview of an end-user enabled model-driven development approach for interactive applications based on annotated services. Proceedings of the 4th Workshop on Emerging Web Services Technology WEWST 09 (pp. 19-28). New York, NY: ACM.

Femminella, M. , Francescangeli, R., Giacinti, F., Maccherani, E., Parisi, A., & Reali, G. (2009). Scalability and performance evaluation of a JAIN SLEE-based platform for VoIP services. Teletraffic Congress, 2009. ITC 21 2009. 21st International, (pp.1-8). Piscataway, NJ: IEEE

Femminella, M., Maccherani, E., & Reali, G. (2010). A software architecture for simplifying the JSLEE service design and creation. *18th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)* (págs. 235-239). Piscataway, NJ: IEEE.

Gonçalves-da-Silva, E., Ferreira-Pires, L., & van-Sinderen, M. (Abril de 2011). Towards runtime discovery,selection and composition of semantic services. *Computer Communication*, *34*, 159-168.

GSMA. (2012). *OneAPI.* Recuperado de Reference Implementation: https://oneapi.aepona.com/

Hussain, S., Wang, Z., Kalil, I., & Diop, A. (2013). Web service testing tools: A comparative study. *IJCSI International Journal of Computer Science*, *10*(1-3), 641-647. Recuperado de http://ijcsi.org/papers/IJCSI-10-1-3-641-647.pdf

Iglesias, C., Fernandez-Villamor, J., del Pozo, D., Garulli, L., & Garcia, B. (2010). Combining domain-driven design and mashups for service development. En *Service Engineering* (pp. 171-200). Viena, Austria: Springer.

Jensen, K., & Kristensen, L. M. (2009). *Coloured petri nets modelling and validation of concurrent systems.* New York, NY: Springer.

Kankanamge, C. (2012). *Web services testing with soapUI.* Birmingham, UK, United Kindom: Packt.

Kecskemeti, G., Terstyanszky, G., Kacsuk, P., & Nemétha, Z. (2011). An approach for virtual appliance distribution for

service deployment. *Future Generation Computer Systems*, 27(3),280-289.

Koning, M., aiSun, C., Sinnema, M., & Avgeriou, P. (2009). VxBPEL: Supporting variability for Web services in BPEL. *Information and Software Technology*, *51*(2), 258–269.

Kryvinska, N., Strauss, C., Auer, L., & Zinterhof, P. (Noviembre de 2008). Conceptual Framework for Services Creation/Development Environment in Telecom Domain. 10th International Conference on Information Integration and Web-based Applications & Services (iiWAS) 2008 (Linz) (pp. 324-331). New York, NY: ACM.

Object Management Group [OMG]. (January de 2012). *UML Profile for Advanced and Integrated Telecommunication Services (TelcoML) [OMG Document Number ptc/2012-01-02]*. Recuperado de http://www.omg.org/spec/TelcoML/

Omelette. (22 de Marzo de 2011). *State-of-the-art in the field of Mashup concepts*. Recuperado de http://www.ict-omelette.eu/documents

Open Cloud. (Marzo de 2009). *Rhino 2.1: Overview and Concepts [Tech. Rep]*. Wellington, Nueva Zelanda: Open Cloud.Recuperadodehttps://developer.opencloud.com/devportal/display/RD/Rhino+Overview+and+Concepts

Oracle. (2012). *Oracle*. Recuperado de Oracle Communications: http://www.oracle.com/us/industries/communications/overview/index.html

Pietschmann, S., Voigt, M., Rümpel, A., & Meißner, K. (2009). CRUISe: Composition of Rich User Interface Services. *Lecture Notes in Computer Science [Web engineering]*(5648), 473–476.

Riabov, A., Bouillet, E., Feblowitz, M., Liu, Z., & Ranganathan, A. (21-25 de Abril de 2008). *Wishful Search: Interactive Composition of Data Mashups*. Obtenido de http://wwwconference.org/www2008/papers/pdf/p775-riabovA.pdf

Rojas-Sierra, G., Estrada-Solano, F., Caicedo, J.A., & Caicedo, O.M. (2011). Technical criteria for value-added services creation, execution and deployment,on next generation networks. 7th Advanced International Conference on Telecommunications (AICT), (pp. 123-129). Red Hook, NY: IARIA

Russell, N., ter Hofstede, A., van der Aalst, W., & Mulyar, N. (2006). *Workflow control-flow patterns: A revised view*. Obtenido de http://www.workflowpatterns.com/documentation/documents/BPM-06-22.pdf

Sánchez, A., Baladrón, C., Aguiar, J., Carro, B., Goix, L.-W., Sienel, J., . . . Bascuñana, A. (2009). User-centric Service Creation and Execution. En E. Di Nitto, A.-M. Sassen, P. Traverso, & A. Zwegers, AT YOUR SERVICE, SERVICE-ORIENTED COMPUTING FROM AN EU PERSPECTIVE (pp. 273-298). Londres, UK, Inglaterra: MIT.

Serrano, C.E. (2008). *Modelo integral para el profesional en ingeniería* [2a ed.]. Popayán, Colombia: Universidad del Cauca

Shevertalov, M., & Mancoridis, S. (2008). A case study on the automatic composition

of network application mashups. *ASE '08 Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering* (págs. 359-362). Washington DC: IEEE Computer Society.

Shin, Y., Yu, C., Chung, S., & Kim, S. (2008). End-user driven service creation for converged service of telecom and Internet. *4th Advanced International Conference on Telecommunications* (pp. 71-76). Los Alamitos, CA: IEEE Computer Society.

SOAP UI. (2013). *SOAPUI*. Recuperado de http://www.soapui.org

Sun Microsystems. (2008). *JAIN SLEE (JSLEE) 1.1 Specification, Final Release [en línea]*. Santa Clara, CA: SUN. Recuperado de http://download.oracle.com/otn-pub/jcp/jain_slee-1_1-final-eval-oth-JSpec/jslee-1_1-fr-spec.pdf

Tao, X., & Wu, J. (27-29 de March de 2010). Research of enterprise application integration based on ESB. *2nd International Conference on Advanced Computer Control (ICACC, 2010) - Beijing* (pp. 90-93). Piscataway, NJ: IEEE.

TelComp 2.0. (2013). *Composición dinámica de Servicios Telco 2.0 [informe técnico]*. Popayán, Colombia: Universidad del Cauca.

Tuchinda, R., Szekely, P., & Knoblock, C. (2008). Building mashups by example. *Proceedings of the 13th international conference on Intelligent user interfaces IUI 08* (pp. 139-148). New York, NY: ACM.

Unmehopa, M., Vemuri, K., & Bennett, A. (2006). *Parlay/OSA : from standards to reality*. Chichester, UK: John Wiley & Sons.

WSO2. (2011). *WSO2 Aplication Server*. Recuperado de http://wso2.com/products/application-server/

Yelmo, J. C., del Álamo, J. M., Trapero, R., & Martín, Y.-S. (2011). Auser-centric approach to service creation and delivery over next generation networks. *Computer Communications, 34*, 209-222.

Yu, J., Falcarin, P., del Álamo, J. M., Sienel, J., Sheng, Q. Z., & Mejia, J. F. (2009). A User-centric mobile service creation approach converging telco and IT services. Eighth International Conference on Mobile Business (pp. 238-242). Los Alamitos, CA: IEEE Computer Society.

Zhu, D., Zhang, Y., Cheng, B., Wu, B., & Chen, J. (Marzo de 2011). HSCEE: A highly flexible environment for hybrid service creation and execution in converged networks. *Journal of Convergence Information Technology, 6*(3), 264-276.

Zuidweg, J. (16 de Marzo de 2009). Middleware en telecomunicaciones. Recuperado el 18 de Noviembre de 2009, de Tecsidel: http://www.tecsidel.es/

Currículum vitae

Julián Andrés Caicedo

M.Sc.(c) en Ingeniería Telemática de la Universidad del Cauca (2014). Participó como investigador en el proyecto TelComp 2.0, *Recuperación y Composición de Componentes Complejos para la Creación de Servicios Telco2.0*, y se desempeñó como Joven Investigador del Programa Jóvenes Investigadores e Innovadores, Colciencias 2011. Realizó una pasantía de investigación dentro del proyecto europeo OMELETTE (*Open Mashups Enterprise service platform for LinkEd data in The TElco domain*) del séptimo programa marco de la Unión Europea para la Investigación de la Universidad Politécnica de Madrid (UPM). Sus áreas de interés son servicios avanzados de telecomunicaciones, y aplicaciones y servicios sobre Internet.

Juan Carlos Corrales

Ingeniero en Electrónica y Telecomunicaciones, y Magíster en Ingeniería Telemática de la Universidad del Cauca; Doctor en Ciencias de la Computación de la Universidad de Versailles Saint-Quentin-en-Yvelines, Francia. En la actualidad es Profesor Titular y Líder del Grupo de Ingeniería Telemática (GIT) de la Universidad del Cauca, Popayán-Colombia.